

Problema 1. Heavy Coins

În regatul lowland a existat odată un zid magnific, care proteja teritoriul acestuia de invadatori timp de secole. Cu toate acestea, cu un an în urmă o bandă de jefuitori a decis că zidul este prea complicat, așa că l-au ars din temelii. Acum, prințesa Maria plănuiește să angajeze constructori iscusiți pentru a îl restaura. Pentru a plăti constructorii, a obținut de la trezorerie $N = 2^K$ monezi, identificate prin numere naturale de la 0 la $N - 1$. Contabilul curții regale a informat-o că sunt două tipuri de monezi – ușoare și grele. Toate monezile grele au aceeași greutate; la fel și toate monezile ușoare. De asemenea, există măcar o monedă ușoară și măcar una grea.

În conformitate cu termenii stabiliți cu constructorii noului zid, prințesa Maria va trebui să îi plătească doar în monede grele. De asemenea, înainte ca treaba să înceapă, trebuie dat un avans de o monedă grea. Problema este că contabilul nu știe câte și care monezi dunt grele. Din fericire, însă, dispune de o balanță cu care poate compara greutatea totală a două mulțimi formate dintr-un număr egal de monezi. El va trebui să se asigure că numărul de monezi grele sunt suficiente pentru plată și să găsească o monedă grea pentru a plăti avansul. Deoarece acest lucru ar putea dura prea mult, iar prințesa Maria vrea ca construcția să înceapă cât mai repede, vă cere ajutorul.

Task

Scrieți un program `coins`, implementând funcția `count_heavy`, care va fi compilată împreună cu programul comisiei și care, prin comunicarea cu acesta va găsi numărul de monezi grele din trezorerie și indicele unei asemenea monezi.

Implementation details

Funcția voastră va trebui să respecte următorul format:

`pair<int, int> count_heavy(int k)`

Va primi ca parametru numărul K (atenție, numărul de monezi este 2^K) și va returna o pereche de două numere întregi – numărul de monezi grele și indicele unei asemenea monezi.

Pentru a facilita comunicarea cu programul comisiei, următoarea funcție vă este pusă la dispoziție:

`int weigh(const vector<int> &a, const vector<int> &b)`

Funcția ia ca parametri doi vectori, fiecare conținând indicii monezilor care vor fi cântărite. Cei doi vectori trebuie să aibă același număr de monezi și să nu conțină numere duplicate. Funcția va returna valoarea:

- 0, dacă cele două mulțimi au aceeași greutate;
- -1, dacă greutatea totală a monezilor din a este mai mică decât greutatea celor din mulțimea b ;
- +1, dacă greutatea totală a monezilor din a este mai mare decât greutatea celor din mulțimea b .

Notă: Complexitatea timp a funcției `weigh` este proporțională cu numărul de elemente din cei doi vectori trimiși ca parametri.

Va trebui să trimiteți sistemului fișierul `coins.cpp`, ce va conține funcția `count_heavy`. Fișierul poate conține alt cod și funcții necesare rezolvării problemei, dar trebuie să nu conțină funcția `main`. De asemenea, nu aveți voie să citiți din citirea standard sau să afișați la consolă. Programul trebuie de asemenea să includă fișierul header `coins.h`, prin scrierea directivei de preprocesor:

```
#include "coins.h"
```

Constrângeri

$1 \leq K \leq 20$

Punctare

Dacă, în timpul execuției, soluția dă răspunsul corect în mai puțin de 10 000 de apeluri ale funcției *weigh*, un coeficient va fi calculat pentru fiecare test conform cu formula $coef = \min\left(1, \frac{11}{9} * \left(\frac{author+1}{yours+1}\right)\right)$, unde *author* este numărul de apeluri ale funcției *weigh* făcute de una din soluțiile autorilor, iar *yours* este numărul de apeluri ale funcției *weigh* făcute de soluția voastră. Altfel, valoarea lui *coef* pentru acel test va fi 0. **Se garantează că pentru toate testele valoarea *author* este strict mai mică decât 1 000.**

Fiecare subtask este format dintr-un număr de grupe, fiecare grupă având exact trei teste. Fiecare grupă este evaluată **independent**, iar numărul de puncte pe care le veți obține pentru grupă este egal cu produsul numărului de puncte asociat grupei și valoarea minimă *coef* pentru testele din acea grupă.

Subtasks

Subtask	Puncte	Alte restricții
1	5	$K \leq 5$
2	10	Există exact o monedă grea sau una ușoară.
3	20	Sunt maxim K monezi grele sau maxim K monezi ușoare.
4	25	$K \leq 10$
5	40	Nicio constrângere suplimentară.

Exemplu de interacțiune

Funcția concurentului	Programul comisiei
	Apelează <code>count_heavy(2)</code>
Apelează <code>weigh({0, 1}, {2, 3})</code>	Returnează -1
Apelează <code>weigh({1, 2}, {0, 3})</code>	Returnează +1
Apelează <code>weigh({1}, {2})</code>	Returnează 0
Returnează {3, 2}	

Testare locală

Pentru testare locală, veți dispune de fișierele `coins.h` și `Lgrader.cpp`. Puneți fișierele `coins.cpp` și cele două fișiere în același folder. Mai apoi, compilați cele trei fișiere împreună. Astfel, veți obține un program care verifică corectitudinea funcției voastre. Programul va citi de la tastatură următoarele date:

- Pe prima linie: un număr natural pozitiv reprezentând valoarea lui K .
- Pe a doua linie: 2^K numere naturale cu valorile 1 sau 0 (reprezentând dacă moneda corespunzătoare este grea sau ușoară).

Programul va afișa la tastatură informații despre apelurile funcției *weigh*, răspunsul returnat de funcția *count_heavy* și un mesaj care indică dacă răspunsul este corect sau nu.